REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
		final technical

4. TITLE AND SUBTITLE

Automatic Verification and Synthesis of Finite-state Hard Real-Time Systems

5. FUNDING NUMBERS

G: N00014-91-J-1901

6. AUTHOR(S)

David Dill

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Computer Science Dept.
Stanford University
Stanford, CA 94305-2140

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of Naval Research
800 North Quincy Street
Arlington, VA 22217-5000

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

14. SUBJECT TERMS

15. NUMBER OF PAGES
6

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
unclassified	unclassified	unclassified	unlimited

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to ***stay within the lines*** to meet ***optical scanning requirements.***

Block 1. Agency Use Only *(Leave blank)*.

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	PR	- Project
G	- Grant	TA	- Task
PE	- Program Element	WU	- Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of...; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

- DOD - See DoDD 5230.24, "Distribution Statements on Technical Documents."
- DOE - See authorities.
- NASA - See Handbook NHB 2200.2.
- NTIS - Leave blank.

Block 12b. Distribution Code.

- DOD - Leave blank.
- DOE - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
- NASA - Leave blank.
- NTIS - Leave blank.

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only)*.

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

David L. Dill
Stanford University
(415) 725-3642
dill@cs.stanford.edu
Automatic Verification and Synthesis of Hard Real-Time Systems
Contract Number N00014-91-J-1901
Reporting Period: 30 September 1993 – 31 May 1994

1 Productivity Measures

These are for the last eight months of the project
Refereed papers submitted but not yet published: 0
Refereed papers published: 0
Unrefereed reports and articles: 1
Books or parts thereof submitted but not yet published: 0
Books or parts thereof published: 0
Patents filed but not yet granted: 0
Patents granted (include software copyrights): 0
Invited presentations: 1
Contributed presentations: 0
Honors received: 0
Prizes or awards received (Nobel, Japan, Turing, etc.): 0
Promotions obtained: 0
Graduate students supported $\geq$ 25% of full time: 1
Post-docs supported $\geq$ 25% of full time: 0
Minorities supported: 0

David L. Dill
Stanford University
(415) 725-3642
dill@cs.stanford.edu
Automatic Verification and Synthesis of Hard Real-Time Systems
Contract Number N00014-91-J-1901
Reporting Period: 30 September 1993 – 31 May 1994

2 Detailed Summary of Technical Progress

As modern control systems become more complex, bugs in human design become increasingly hard to detect by traditional methods such as simulation and prototype testing. The introduction of *timing information* into specifications makes analyzing processes even more subtle. Thus there is great need for formal methods for proving the correctness of real-time systems. A related topic of research is how to automatically generate provably-correct real-time processes, thus bypassing the need for debugging and iterative design. The goals of this project are to develop computationally feasible automatic methods for the formal verification and synthesis of hard real-time systems.

In previous years, we have investigated algorithms based on generalizations of finite-state minimization for verification and the use of approximations of various kinds. We have also explored formalisms for supervisory synthesis under real-time constraints. In this, the final year of the project, we have focussed our efforts on making approximation methods work for more realistic system designs. We have also invested considerable effort in generalizing our approximation techniques in the hope that they can be used in other domains.

Our current algorithm works by successive approximation. It proceeds in a sequence of forward and backwards passes. In each pass, it maintains both an overapproximation and an underapproximation of the reachable state space. If no "bad states" appear in the overapproximation, or if a bad state appears in the underapproximation, the verifier can halt immediately with the correct result. Otherwise, it reverses direction and refines the approximations to increase accuracy. Ultimately, the verifier will always halt with the correct result (unless it runs out of memory or the user runs out of patience). There are several novel ideas in the approximation scheme.

The verifier uses a hybrid symbolic representation of the state space, consisting of sets of linear inequalities of the form $x - y < c$, where c is an integer, to represent timing, and a binary decision diagram representing sets of control states (but not representing the timing).

This year, we have cleaned up and improved the efficiency of the implementation of our verifier significantly. We added "invariants" to states, which allow us to express upper bounds as well as lower bounds on delays. Also, we have added "urgent" actions, which happen as soon as they are enabled.

We are now consistently able to handle all of the examples we set out to do, including the difficult ethernet example of Weinberg and Zuck (from the Concur 92 conference).

In addition, we have an algorithm for handling "skewed clock automata," which allow timers that increase at variable rates, where only upper and lower bounds on the rates are known. We have found a restricted but useful class of these automata which can be analyzed exactly by converting into ordinary timed automata. Using this result, we were able to verify completely automatically a protocol for "Manchester encoding" that was previously done by hand using timed automata.

Most of these results are contained in Howard Wong-Toi's PhD thesis, which will be completed within a month.

David L. Dill
Stanford University
(415) 725-3642
dill@cs.stanford.edu
Automatic Verification and Synthesis of Hard Real-Time Systems
Contract Number N00014-91-J-1901
Reporting Period: 30 September 1993 – 31 May 1994

3 Lists of publications, presentations and reports

3.1 Published

Howard Wong-Toi and David L. Dill, "Approximations for Verifying Timing Properties," Chapter 7 in Theories and Experiences for Real-Time System Development (Proceedings First Amast Workshop on Real-Time System Development, 1993), Teo Rus and Charles Rattray (Ed), World Scientific Publishing, 1994.

David L. Dill
Stanford University
(415) 725-3642
dill@cs.stanford.edu
Automatic Verification and Synthesis of Hard Real-Time Systems
Contract Number N00014-91-J-1901
Reporting Period: 30 September 1993 – 31 May 1994

4 Transitions and DoD interactions

None.

David L. Dill
Stanford University
(415) 725-3642
dill@cs.stanford.edu
Automatic Verification and Synthesis of Hard Real-Time Systems
Contract Number N00014-91-J-1901
Reporting Period: 30 September 1993 - 31 May 1994

5 Software prototypes

We have a prototype of the approximation-based verifier for timed automata, along with various examples.